We Claim:

1. A method of determining the performance of a computer program when said program is executed on a multiple processor computer system having a shared resource, said program producing multiple parallel processes which can be executed in parallel with other processes and multiple serial processes which can execute in parallel only with parallel processes, said resource being shared between multiple processes such that said computer system implements a rollback scheme to arbitrate between processes which compete for access to said resource, said method comprising:

a) determining how many parallel processes complete their assigned tasks (Lp);

b) determining how many serial processes complete their assigned tasks (Ln);

c) determining how much processing time is used by parallel processes which complete their tasks (Up);

d) determining how much processing time is used by serial processes which complete their tasks (Un);

e) determining how may parallel processes have not been able to complete their tasks due to a first denial of access to the shared resource, said first denial of access being caused by a serial process (Rpn);

f) determining how many parallel processes have not been able to complete their tasks due to a second denial of access to the shared resource, said second denial of

access being caused by another parallel process (Rpp);

g) determining how many serial processes have not been able to complete their tasks due to a third denial of access to the shared resource, said third denial of access being caused by a parallel process (Rnp);

h) determining how much processing time is spent by serial processes while waiting to finish its tasks, said waiting being caused by other serial processes finishing their tasks (W);

i) determining how much processing time is wasted by parallel processes which are not able to complete their tasks (COHp);

j) determining how much processing time is wasted by serial processes which are not able to complete their tasks (COHn); and

k) calculating a probability of a process not being able to complete its tasks due to competition for said resource, said probability being calculated using data gathered in steps a) - j),

wherein,

said rollback scheme comprises:

- determining between two or more processes competing for access to said resource which process gives access to said resource; and
- causing processes which have not been granted access to said resource to discard results which have been previously obtained by said processes which have not been granted access.

2. A method as in claim 1 further including inserting in said program substantives which gather said data gathered in steps a) - j).

3. A method as in claim 1 wherein said data gathered in steps a) - j) is sent to a second computer system and step k) is executed at said second computer system.

4. A method as in claim 1 wherein step k) is calculated using the formula,

$$Qn\,p = \frac{Rpn + Rn\,p}{max(Ln\,p, Lpn) + Rpn + Rn\,p}$$

where,

max(Lnp, Lpn) is an estimate of the number of non-colliding parallel/serial process pairs;

Rpn is the number of parallel processes which have not been able to complete their tasks due to a denial of access to said resource caused by a serial process;

Qnp the probability of a process not being able to complete its tasks when a parallel process and a serial process compete for said resource;

Rnp is the number of serial processes which have not been able to compete their tasks due to a denial of access to said resource caused by a parallel process;

Lnp is the number of successful executions of parallel processes running against

successful executions of serial processes; and

Lpn is the number of successful executions of serial processes running against successful executions of parallel processes.

5. A method as in claim 1 wherein step k) is calculated using the formula

$$Qpp = \frac{Rpp}{0.5Lpp + Rpp}$$

where,

Qpp is the probability of a process not being able to complete its tasks when a parallel process competes with another parallel process for said resource;

Rpp is the number of parallel processes in contention for said resource; and

Lpp estimates the number of successful executions of parallel processes when running against other parallel process.

6. A method as in claim 5 wherein

$$Lpp = \frac{Upp}{Up \,/\, Lp}$$

where,

Upp is the total time spent on successful executions of parallel processes when run against other parallel processes;

Up is the total processing time spent by parallel processes which complete their tasks; and
Lp is the number of parallel processes which complete their tasks.

7. A method as in claim 4 wherein Lpn is calculated using the formula:

$$Lpn = \frac{\text{Un - W - Dpn}}{(Up \ / \ Lp)}$$

where,
Up is the total processing time spent by parallel processes which complete their tasks;
Lp is the number of parallel processes which complete their tasks;
Un is how much processing time is used by serial processes which compete their tasks;
W is the processing time spent by serial processes while waiting to finish its tasks, said waiting being caused by other serial processes finishing their tasks; and
Dpn is the time wasted by parallel processes due to contention for said resource with serial processes.

8. A method as in claim 7 wherein Dpn is calculated using the formula,

$$\text{Dpn} = \frac{(\text{COH p} * \text{Rpn})}{\text{Rpn} + \text{Rpp}}$$

where

Rpn is the number of parallel processes which have not been able to complete their tasks due to a denial of access to said resource caused by a serial process;

Rpp is the number of parallel processes in contention for said resource;

COHp is the amount of processing time wasted by parallel processes which have not been able to complete its tasks.

9. A method as in claim 6 wherein the total time spent on successful executions of parallel processes when run against other parallel processes is calculated using the formula,

Upp=Up-(Un+COHn-W-Dpn)-Dpp

where

Upp is the total time spent on successful executions of parallel processes when run against other parallel processes;

Up is the total processing time spent by parallel processes which complete their tasks;

Un is how much processing time is used by serial processes which compete their tasks;

COHn is the amount of processing time wasted by serial processes which have not been able to complete its tasks;

W is the processing time spent by serial processes while waiting to finish its tasks, said waiting being caused by other serial processes finishing their tasks;

Dpn is the time wasted by parallel processes due to contention for said resource with serial processes; and
Dpp is the time wasted by parallel processes due to contention for said resource with other parallel processes.

10. A method as in claim 9 wherein Dpp is calculated using the formula,

$$Dpp = COHp - \left(\frac{(COHp + Rpn)}{Rpn + Rpp} \right)$$

where

Dpp is the time wasted by parallel processes due to contention for said resource with other parallel processes;
COHp is the amount of processing time wasted by parallel processes which have not been able to complete its tasks;
Rpn is the number of parallel processes which have not been able to complete their tasks due to a denial of access to said resource caused by a serial process; and
Rpp is the number of parallel processes in contention for said resource.

11. A method of determining the effect on program performance of resource access contention between processes produced by a computer program executed on a multiple processor computer system having a resource shared among said processes, said method comprising:

a) inserting multiple subroutines in said program, said substantive measuring data relating to the access of said processes to said resource and the effect of said resource to the execution time and number of said processes;
b) gathering said data measured by said subroutines; and
c) calculating a probability that contention between processes for said resource will result in wasted processing time, said probability being based on data gathered in step b).

12. A method as in claim 11 wherein said data gathered in step b) includes,
a) the number of parallel processes which complete their assigned tasks (Lp);
b) the number serial processes which complete their assigned tasks (Ln);
c) processing time used by parallel processes which complete their tasks (Up);
d) processing time used by serial processes which complete their tasks (Un);
e) the number of parallel processes which have not been able to complete their tasks due to a first denial of access to the shared resource, said first denial of access being caused by a serial process (Rpn);
f) the number of parallel processes which have not been able to complete their tasks due to a second denial of access to the shared resource, said second denial of

access being caused by another parallel process;

g) the number of serial processes which have not been able to complete their tasks due to a third denial of access to the shared resource, said third denial of access being caused by a parallel process (Rnp);

h) processing time used by serial processes while waiting to finish its tasks, said waiting being caused by other serial processes finishing their tasks (W);

i) processing time used wasted by parallel processes which are not able to complete their tasks (COHp); and

j) processing time which is wasted by serial processes which are not able to complete their tasks (COHn).

13. A method as in claim 12 wherein said computer system implements a rollback scheme to arbitrate between processes competing for said resource, said rollback scheme comprising:

- determining between two or more processes competing for access to said resource which process gives access to said resource; and

- causing processes which have not been granted access to said resource to discard results which have been previously obtained by said processes which have not been granted access.

14. A method as in claim 1 wherein said multiple processes complete for write access to said shared resource.

15. A method as in claim 11 wherein said multiple processes complete for write access to said shared resource.